System Commands

This section contains a detailed explanation of each Natural system command.

- Command Syntax
- Issuing Commands

Alphabetical List of System Commands

Command Syntax

The following symbols are used within the syntax descriptions of system commands:

ABCDEF	Upper-case letters in boldface indicate that the term is either a Natural keyword or a Natural reserved word that must be entered exactly as specified.
ABCDEF	If an optional term in upper-case boldface letters is completely underlined, this indicates that the term is the default value. If you omit the term, the underlined value applies.
ABCDEF	If a term in upper-case boldface letters is partially underlined, this indicates that the underlined portion is an acceptable abbreviation of the term.
abcdef	Lower-case letters in italics are used to represent variable information. You must supply a valid value when specifying this term.
[ ]	Elements contained within square brackets are optional. If the square brackets contain several lines stacked one above the other, each line is an optional alternative. You may choose at most one of the alternatives.
{ }	If the braces contain several lines stacked one above the other, each line is an alternative. You must choose exactly one of the alternatives.
|	The vertical bar separates alternatives.
...	A term preceding an ellipsis may optionally be repeated. A number after the ellipsis indicates how many times the term may be repeated. If the term preceding the ellipsis is an expression enclosed in square brackets or braces, the ellipsis applies to entire bracketed expression.
,...	A term preceding a comma-ellipsis may optionally be repeated; if it is repeated, the repetitions must be separated by commas. A number after the comma-ellipsis indicates how many times the term may be repeated. If the term preceding the comma-ellipsis is an expression enclosed in square brackets or braces, the comma-ellipsis applies to entire bracketed expression.
:...	A term preceding a colon-ellipsis may optionally be repeated; if it is repeated, the repetitions must be separated by colons. A number after the colon-ellipsis indicates how many times the term may be repeated. If the term preceding the colon-ellipsis is an expression enclosed in square brackets or braces, the colon-ellipsis applies to entire bracketed expression.
Other symbols (except [ ] { } | ... ,... :...)	All other symbols except those defined in this table must be entered exactly as specified. **Exception**: The SQL scalar concatenation operator is represented by two vertical bars that must be entered literally as they appear in the syntax definition.

Example of Command Syntax:

CATALOG [*object-name* [*library-id*]]

- **CATALOG** is a Natural keyword which you must enter as specified. The underlining indicates that you may also enter it in abbreviated form as **CAT**.
- *program-name* and *library-id* are user-supplied operands for which you specify the name of the program you wish to deal with and the ID of the library in which that program is contained.
- The square brackets indicate that *program-name* and *library-id* are optional elements which you can, but need not, specify. The grouping of the brackets indicate that you can specify CATALOG alone, or CATALOG followed either by a program name only or by a program name and a library ID; however, you cannot specify a library ID if you do not also specify a program name.

Issuing Commands

Commands can be issued from the Direct Command line of the Natural main menu or from the command lines of the program editor, map editor, or data area editor.

CATALL

```
CATALL [object-name]
```

This command is used to catalog all objects in the current library.

The CATALL command invokes the following window:

```
+-------------------- Start Catall ---------------------+
| Object Name: *                                        |
|                                                       |
| Recatalog only if module exists                 X     |
| Catalog all source objects                            |
|                                                       |
| DDMs                    X  Helproutines         X     |
| Global Data Areas       X  Maps                 X     |
| Local Data Areas        X  Programs             X     |
| Parameter Data Areas    X  Generate new map source    |
| External Subroutines    X  Classes              X     |
| Subprograms             X                             |
|                                                       |
+-------------------------------------------------------+
```

In this window, you can make the following specifications:

- **Object name**
  If you wish CATALL to be performed for all objects of the selected types in the current library, specify "*" as object name.
  If you wish it to be performed for a certain range of objects, you can use asterisk notation   for the name.
- **Recatalog only if module exists**
  If you mark this field, only those objects for which object modules already exist in the current library will be cataloged again; objects which only exist in source form will not be cataloged; or:
- **Catalog all source objects**
  If you mark this field, *all* selected objects will be cataloged.
- **Object types**
  By default, CATALL applies to objects of all types in the current library (all object types are marked with "X").
  If you wish objects of a certain type *not* to be affected by CATALL, overwrite the respective "X" with a blank.
- **Generate new map source**
  Maps created with previous Natural versions are not necessarily compatible with Natural Version 3.1 and above. Mark this field to ensure that maps are converted during the catalog operation. This option converts and stores maps in source *and* object form.

During CATALL processing, a statistics window appears and the objects being cataloged are listed.

Note:
If you press any key while CATALL processing is in progress, CATALL will be stopped.

Upon successful completion of processing, an information message is displayed.

If an object was not cataloged successfully, a window showing the object name, error number and error line is displayed.

CATALOG

```
CATALOG [object-name [library-id] ]
```

The CATALOG command is used to compile the Natural programming object currently in the source work area of an editor and store the resulting object module.

object-name

As *object-name*, you specify the name under which the object is to be cataloged.

If you do not specify an *object-name*, the object will be cataloged under the name as set by the last command which caused a source object to be read into the editor (for example, EDIT, READ, RUN).

library-id

If you do not specify a *library-id*, the object will be cataloged in the current library. If you wish the object is to be cataloged into another library, you must specify the *library-id* of that library.

Under Natural Security, you cannot specify a *library-id*; that is, you can store an object only in your current library.

Note:
The CATALOG command cannot be used if the RECAT profile parameter has been set to ON; in this case, use the STOW command to ensure that source code and object code match.

CHECK

```
CHECK
```

The CHECK command is used to verify the syntax of a source program.

If an error is detected, syntax checking is suspended and the error line is displayed. You may then either correct the line (whereupon the checking will be continued) or you press ENTER without modifying the displayed line. This stops checking and invokes the editor, where you can correct the error.

Syntax checking is also performed as part of the RUN, STOW and CATALOG commands.

CLEAR

```
CLEAR
```

The CLEAR command clears the source work area. This command must be used if a new program is to be created and there is another object in the source work area.

COMPOPT

```
COMPOPT [option = value]
```

This command is used to set various compilation options. The options are evaluated when a Natural programming object is compiled.

When you issue only the COMPOPT command itself, a screen will be displayed on which you can set the options described below.

Note:
The default values of the individual options are set with the respective profile parameters in the Natural parameter module.

Interpretation of Database Short Field Names (DBSHORT):

ON	Database field names in programming objects are considered long names (as defined in the corresponding DDM) - except 2-character field names, which are considered short names (as used by the underlying database system).
OFF	All database field names in programming objects are considered long names, regardless of their length. This avoids possible misinterpretations of database field names in programs.

Generation of Global Format IDs (GFID):

This option allows you to control Natural's internal generation of global format IDs so as to influence Adabas's performance concerning the re-usability of format buffer translations.

ON	Global format IDs are generated for all views.
VID	Global format IDs are generated only for views in local/global data areas, but not for views defined within programs.
OFF	No global format IDs are generated.

For details on global format IDs, see the Adabas documentation.

Allow Old Version 2.2 Syntax (V22COMP):

Note:
This option will be available only for a limited period of time to allow a smooth transition to Version 3.1 syntax and above. It will be removed again with a subsequent release of Natural.

The following inconsistent syntax construction not intercepted by Version 2.2 lead to a syntax error with Version 3.1 and above:

DEFINE DATA: inconsistent number of decimal digits in constant value.

To allow you a smooth transition from Version 2.2 to Version 3.1, you can use this option.

ON	The above syntax constructions will *not* lead to a syntax error. Thus you are able to compile your existing programs under Version 3.1 until you have adjusted them to the Version 3.1 requirements.
OFF	The above syntax constructions will lead to a syntax error.

For details on the above inconsistencies, please refer to your Natural Version 4.1.2 Release Notes.

option=value

Instead of changing an option on the screen, you can also specify it directly with the COMPOPT command. The keywords for the individual options are shown in parentheses (on the COMPOPT screen and in the above description).

Example:

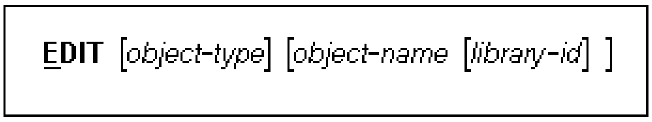
COMPOPT DBSHORT=ON

DEBUG

This command is used to invoke the debugging utility. This utility is described in the Natural Debugging Documentation.

EDIT

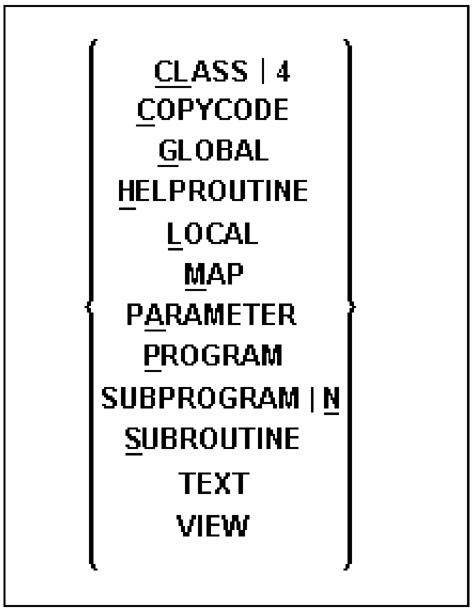
Syntax 1



```
EDIT [object-type] [object-name [library-id] ]
```

The EDIT command is used to invoke a Natural editor for the purpose of editing a Natural object.

object-type



```
CLASS | 4
COPYCODE
GLOBAL
HELPROUTINE
LOCAL
MAP
PARAMETER
PROGRAM
SUBPROGRAM | N
SUBROUTINE
TEXT
VIEW
```

EDIT VIEW only works in the current library and when an *object-name* is specified.

If you are in the program editor and enter EDIT object-name (whereas the Natural object is of type copycode, program, subprogram,subroutine, class, text, helproutine), a parallel edit session will be opened. See also the Program Editor, editor command NEXT.

object-name

As *object-name* you specify the name of the object to be edited (maximum 8 characters; exception: view names maximum 32 characters); Natural will then load the object into the edit work area of the appropriate editor and set the object name for a subsequent SAVE, CATALOG or STOW command.

If you do not specify an *object-name* and there is no object in the edit work area, the empty program editor screen will be invoked where you can create a program.

library-id

As *library-id*, you specify the library in which the object to be edited is contained. The *library-id* need only be specified if the object to be edited is contained in a library other than the one you are currently logged on to.

Under Natural Security, you cannot specify a *library-id*; that is, you can edit only objects which are stored in your current library.

Syntax 2

EDIT FUNCTION *subroutine-name*

The EDIT FUNCTION command may be used to edit a subroutine using the subroutine name (not the object name).

Example:

```
   DEFINE SUBROUTINE ABC
   ...
   END-SUBROUTINE
   END
```

Assuming that the above subroutine has been saved under the object name "SUB", you may edit subroutine ABC either by issuing the command EDIT S SUB or by EDIT F ABC.

EXECUTE

```
{EXECUTE [REPEAT] program-name [library-id]     }
{                 program-name [parameter...]   }
```

The EXECUTE command is used to execute a Natural object module.

The object module must have been cataloged (that is, stored in object form) in a Natural library or linked to the Natural nucleus.

The keyword EXECUTE is optional; it is sufficient to specify the name of the program to be executed. The execution of an object module does not affect the source program currently in the editor work area.

If the program being executed produces multiple screen output and you wish the screens to be output one after another without intervening prompts, specify the keyword REPEAT together with the keyword EXECUTE.

Note:
When entered in the command line of the program editor, the system command EXECUTE must not be abbreviated to EX, as the program editor would interpret this as the editor command EX (see also the section The Program Editor).

program-name

The name of the program to be executed. If you do not specify a *library-id*, Natural can only execute the specified program if it is stored either in your current library or in a steplib library (the default steplib is "SYSTEM").

library-id

If the program is stored in another library, specify the *library-id* of that library. In this case, Natural can execute the program only if it actually stored in the specified library.

Under Natural Security, you cannot specify a *library-id*; that is, you can execute only programs which are stored in your current library.

Passing Parameters to the Program

When a Natural program is executed by specifying the program name only, input parameters may be passed to the Natural program. These parameters will be read by the first INPUT statement in the executed Natural program. The parameters may be specified as either positional parameters or keyword parameters with each value separated by the input delimiter character.

If one of the parameters is a string which contains blankes, the tranfer will only be executed if directly after the program name an input delimiter is set.

Examples of EXECUTE Command:

EXECUTE PROG1

EXECUTE PROG1 ULIB1

PROG1

PROG1 VALUE1 VALUE2 VALUE3

PROG1 VALUE1, VALUE2, VALUE3

PROG1 PARM1=VALUE1, PARM2=VALUE2, PARM3=VALUE3

PROG1 PARM3=VALUE3 PARM1=VALUE1 VALUE2

PROG1, ab cd ef, gh, de fg, ab

FIN

```
FIN
```

The FIN command is used to terminate a Natural session.

Copyright Software AG 2001

GLOBALS

```
GLOBALS [parameter...]
```

The GLOBALS command is used to set Natural session parameters.

The GLOBALS command corresponds to the SET GLOBALS statement. The parameters are the same as for the SET GLOBALS statement. The GLOBALS command and SET GLOBALS statement can both be used in the same Natural session. Parameter values specified with a GLOBALS command remain in effect until they are overridden by a new GLOBALS command or SET GLOBALS statement, the session is terminated, or you log on to another library.

Parameter values can be supplied in any order and must be separated by a blank. If more parameters are specified than will fit on one command line, several GLOBALS commands must be issued.

If the GLOBALS command is entered without parameters, a dialog window is displayed from which you can select a subset of possible parameters and modify the values.

The individual parameters are described in the section Session Parameters of the Natural Reference Documentation.

Some parameter values may be overridden during program execution using the LIMIT, EJECT, and FORMAT statements and by format entries specified in INPUT, DISPLAY, PRINT, and WRITE statements.

HELP

```
⎧HELP⎫  ⎡[NAT] nnnn⎤
⎩ ?  ⎭  ⎣USER  nnnn⎦
```

The HELP command invokes online help for error messages.

The following options are available with the HELP command:

HELP

If you enter only the command HELP, a window is displayed in which you enter the number of the desired message, and specify if it is a Natural system message or a user message of the current library.

HELP [NAT]*nnnn*

If you enter HELP and a number (up to 4 digits, optionally prefixed by "NAT"), the detailed message text for the Natural error condition associated with that number will be displayed, i.e., the long text of the Natural *system* error message NAT*nnnn.*

HELP USER*nnnn*

"HELP USER*nnnn*" or "HELP U*nnnn*" displays the long text of the *library-specific* message *nnnn* in the current library.

INPL

```
INPL [R]
```

If you enter the INPL command without any parameters, the INPL utility will be invoked. This utility is *only* used for the loading of Software AG installation datasets into the file system as part of the installation process (as described in the Installing and Setting Up Natural on UNIX or Installing and Setting Up Natural on OpenVMS Documentation).

Apart from that, you use the utility NATLOAD in library SYSUNLD (see the Natural NATUNLD/NATLOAD Utilities documentation) to load objects into the file system.

In addition, the INPL command provides the following special function:

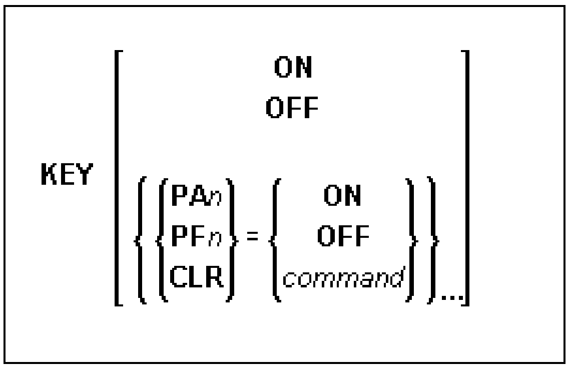
INPL R - Natural Security Recover

This option is only available if Natural Security is installed.

"INPL R" is used to reset the access to the Natural Security library "SYSSEC" to its original status.

If you issue "INPL R", the user "DBA", the library "SYSSEC", and the link between the two will be redefined as after the initial installation, while all other links to "SYSSEC" will be cancelled. See also Inaccessible Security Profiles in the section Countersignatures of the Natural Security documentation.

KEY



With the KEY command, you can assign functions to keys on the keyboard of video terminals. Moreover, you can change, activate and deactivate the assigned functions.

This is possible for the following keys: the PA keys PA1 to PA3, the PF keys PF1 to PF24, and the CLEAR key.

To each of these keys you can assign one of the following functions:

- a Natural system command,
- a Natural terminal command,
- a user-defined command.

Natural will execute the assigned command whenever you press the corresponding key in command mode (Direct Command window).

Note:
Assignments made with the system command KEY are totally independent of assignments made with a SET KEY statement in a program.

Function-key assignments can also be made by the Natural administrator via the profile parameter KEY.

Assigning Commands

If you enter only the command **KEY** (without parameters), the "Function-Key Assignments" screen will be displayed. On this screen, you can assign commands to the individual keys.

To assign a different command to a key, you overwrite the existing entry.

To delete an command assignment, you delete the entry or overwrite it with blanks.

You can also assign commands to individual keys by specifying them directly with the KEY command; that is, you specify **KEY***key*=*command* (*key* being the name of the key and *command* the name of the command that is to be assigned to the key). For example:

KEY PF1=CLEAR

If the assigned command contains blanks, it has to be enclosed in apostrophes. For example:

KEY PF13='UPDATE OFF'

Activating/Deactivating All Keys - KEY ON/OFF

With the command **KEY OFF** you deactivate all function-key assignments: if you press a function key, Natural will return an appropriate message indicating that the key is not active.

With the command **KEY ON** you re-activate all function-key assignments that have previously been deactivated with KEY OFF.

You can also activate/deactivate the keys by overwriting the entry ON/OFF in the field "Activate Keys" at the top right-hand corner of the "Function-Key Assignments" screen.

Activating/Deactivating Individual Keys - KEY *key*=ON/OFF

With the command **KEY***key***=OFF** you deactivate the command assigned to a specific *key*. For example:

KEY PF24=OFF

With the command **KEY***key***=ON** you re-activate a previously deactivated command assignment. For example:

KEY PF24=ON

When you deactivated an individual key (for example PF24=OFF), then deactived all keys (KEY=OFF) and then activate all keys again (KEY=ON), the individually deactivated key is activated, too.

LAST

```
        [ -   ]
LAST    [ -nn ]
        [ *   ]
        [ ?   ]
```

With the LAST command, you can have the command(s) that was/were last executed displayed. Moreover, you can have the displayed command(s) executed again.

Only system commands that you actually entered can be displayed via the LAST command; commands issued internally by Natural as a result of a command you entered are not available via LAST.

The LAST command provides the following options:

LAST

The system command that was issued last is placed in the Direct Command window and can be executed.

LAST -

The system command that was issued last is placed in the Direct Command window and can be executed.

If you enter "LAST -" again, the last but one command will be placed in the Direct Command window. By repeatedly entering "LAST -", you can thus "page" backwards command by command. (Instead of repeatedly entering it by hand, you can assign "LAST -" to a PF key via the system command KEY.)

LAST *-nn*

The *nn*th previous system command is written into the Direct Command window and can be executed.
Natural stores the previous 20 commands issued; *nn* must therefore not be greater than 20.

When a command is placed in the Direct Command window, you can execute it again by pressing ENTER. You can also overwrite it before you execute it again.

LAST *

If you enter "LAST *", a window will be displayed showing the last 20 commands that were issued. Use PF8 and PF7 to scroll forward and backward if more than 10 commands are displayed. In the window, you can select the command(s) you wish to be executed again:

- To execute *a single command* again, either mark the command with the cursor and press F5, or mark the command with any character and press ENTER.
- To execute *several commands* again, mark them with numbers in the order in which you wish them to be executed and press ENTER; the commands will then be executed in ascending order of numbers.

Before you have a command executed again, you can overwrite it.

LAST ?

With "LAST ?" you can call the Help function for the LAST command.

LASTMSG

```
LASTMSG
```

With the LASTMSG command, you can display additional information about the error situation which has occurred last.

When Natural displays an error message, it may in some cases be that this error is not the actual error, but an error caused by another error (which in turn may have been caused by yet another error, etc.) In such cases, the LASTMSG command allows you to trace the issued error back to the error which has originally caused the error situation.

When you enter the command LASTMSG, you will get - for the error situation that has occurred last - the error message that has been displayed, as well as all preceding (not displayed) error messages that have led to this error.

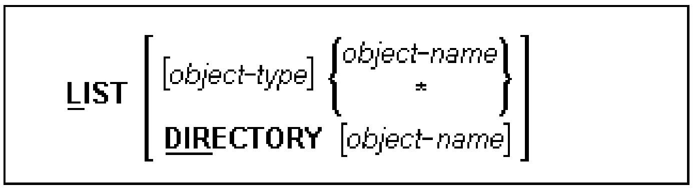
When you mark one of these messages with the cursor, you will get the following information on the corresponding error:

- error number;
- number of the line in which the error occurred;
- name, type and level of the object that caused the error;
- name, database ID and file number of the library containing the object;
- error class (system = error issued by Natural; user = error issued by user application);
- error type (runtime, syntax, command execution, session termination, program termination, remote procedure call);
- date and time at which the error occurred.

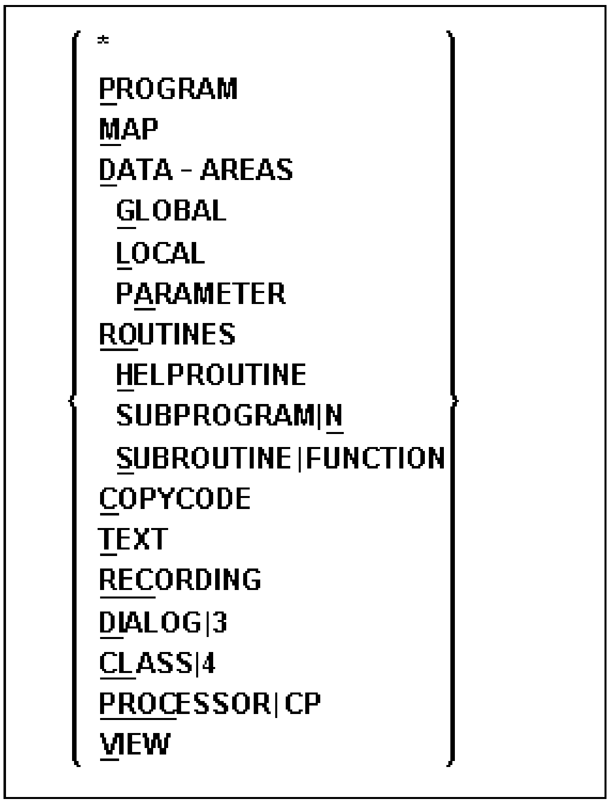
Note:
The library SYSEXT contains a user exit "USR2006" which allows you to display - in your Natural application - the error information supplied by LASTMSG.

Note for Natural RPC:
In the case of an error on the server, the following error information is not displayed: database ID, file number, date and time.

LIST



object-type



The LIST command is used to list one or more objects which are contained in the current library.

Displaying an Individual Source

LIST	If you enter only the LIST command itself, without any parameters, the contents of the source work area will be listed.
LIST *object-name*	If you enter a single *object-name* with the LIST command, you need not specify the *object-type*; the object’s source code will be listed.

Displaying Directory Information

LIST DIR	This command displays the directory information about the object currently in the source work area.
LIST DIR *object-name*	This command displays the directory information about the specified object. To display the directory information of several objects, you use asterisk notation for the *object-name*.

LIST *object-type object-name*

If you specify an *object-type*, you must also specify an *object-name*.

To have all objects in the current library listed, you specify "*" for the *object-type*, but no *object-name*.

To have all objects of a certain type listed, you specify a certain *object-type* and "*" for the *object-name*.

If you wish a certain range of objects to be listed, you can use asterisk notation for the *object-name*:

Examples:

LIST * lists all objects in the current library - regardless of their types.

LIST S * lists all subroutines in the current library.

LIST SYS* lists all objects (of any type) whose names begin with "SYS".

LIST M SYS* lists all maps whose names begin with "SYS".

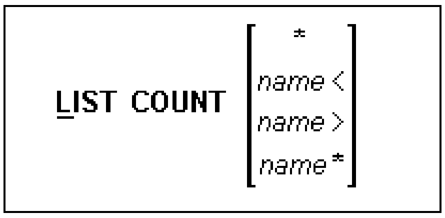
LIST DIR PRG01 lists directory information of object PRG01 in current library.

To select an object from the selection list for a function, you simply mark the object with the appropriate function code in the left-hand column. The function codes are:

Code	Function
C	Check the object’s source code.
D	Read the object’s source code.
E	Edit the object’s source (equivalent to the system command EDIT).
H	Print hardcopy of the object’s source.
L	List the object’s source code.
I	List Directory of the object’s source code.
R	Run (that is, compile and execute) the object’s source (equivalent to the system command RUN).
S	Stow the object in source and object form (equivalent to the system command STOW).
U	Delete the object’s source and object form.
X	Execute the object (equivalent to the system command EXECUTE).
.	End.

Enter "?" or use F2 to display the list of the available function codes for the selected object.

LIST COUNT



The LIST COUNT command is used to list the number of Natural objects in your current library.

The following command options are available:

LIST COUNT	will display the total number of objects.
LIST COUNT *	will display the number of objects broken down by object types.
LIST COUNT*name*<	will display the number of objects whose names are less/equal *name*.
LIST COUNT*name*>	will display the number of objects whose names are greater/equal *name*.
LIST COUNT*name**	will display the number of only those objects whose names begin with *name*.

LIST XREF

This command displays all active cross-reference data for the current library. This command is only available if Predict with active cross-references is installed.

Note:
This command is only available if Predict has been installed.

LOGOFF

```
LOGOFF
```

The LOGOFF command corresponds to the command LOGON SYSTEM, that is, it causes a logon to the library SYSTEM.

LOGOFF has no effect on existing Natural global parameter settings.

For additional information on LOGOFF processing under Natural Security, see your Natural Security Documentation.

LOGON

```
LOGON library-id [password]
```

The LOGON command is used to establish a library ID for the user. In the specified library, all newly created source or object programs saved during the session will be stored (unless you explicitly specify another library ID in a SAVE, CATALOG or STOW command).

The LOGON command has no direct effect on the source code currently in the source work area.

For LOGON processing under Natural Security, ask your Natural Security, administrator or see your Natural Security Documentation.

LOGON causes all Natural global data areas, all assignments made using the SET KEY statement and retained ISN lists to be released. DDMs contained in the DDM buffer area are also released.

Naming Conventions for Libraries

A library ID may be 1 to 8 characters long and must not contain blanks.

A library ID may consist of the following characters:

A - **Z**	upper-case alphabetical characters
0 - **9**	numeric characters
-	hyphen
_	underscore

The first character of a library ID must be an upper-case alphabetical character.

MAIL

```
MAIL mailbox-id
```

This command is available only if Natural Security is installed.

A *mailbox* is a kind of "notice board" used to broadcast messages, which is defined in Natural Security.

With the MAIL command, you can invoke a mailbox to modify its contents and/or expiration date. The *mailbox-id* you specify (maximum 8 characters) must be defined in Natural Security.

See your Natural Security Documentation for details on mailboxes.

NATLOAD

```
NATLOAD
```

This command is used to invoke the NATLOAD utility; see the Natural NATUNLD/NATLOAD Utilities documentation.

NATUNLD

```
NATUNLD
```

This command is used to invoke the NATUNLD utility; see the Natural NATUNLD/NATLOAD Utilities documentation.

PROFILE

```
PROFILE
```

This command is available only if Natural Security is installed.

With the PROFILE command, you can display the security profile currently in effect. This profile informs you of the conditions of use in effect for you in your current Natural environment.

PURGE

```
PURGE [object-name] ...
```

The PURGE command is used to delete one or more source objects.

As *object-name*, you specify the name(s) of the object(s) to be deleted. You can only delete objects which are stored in the library to which you are currently logged on.

If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation for the *object-name*.

If you enter the PURGE command without an *object-name* or without an *object-name* but with an asterisk, a list of all objects in the current library will be displayed; on the list, you may then mark the objects to be deleted.

READ

```
READ object-name [library-id]
```

The READ command is used to transfer an object which is stored in source form into the source work area. Any object currently in the source work area will be overlaid by the object read.

object-name

The name of the object to be read. If *object-name* is specified without a library ID, the object will be read only if it is stored in the library to which you are currently logged on.

library-id

The library in which the object to be read is contained. If both *object-name* and *library-id* are specified, Natural will only read the object if it is stored under the specified library ID.

Under Natural Security, you cannot specify a *library-id*; that is, you can read only objects which are stored in your current library.

REGISTER

This command is used in conjunction with NaturalX. It is described in the NaturalX Documentation.

RENUMBER

```
RENUMBER [(n)]
```

The RENUMBER command is used to renumber the lines in the source program currently in the source work area.

n may be used to specify the increment to be used for renumbering. The default is 10.

Note:
Each time a source is READ into the source area it will be renumbered to the most optimum number space.

RETURN

```
        [ I  ]
RETURN  [ nn ]
        [ *  ]
```

The RETURN command may be used to return to a previous (or initial) Natural application.

If RETURN is specified without any parameters, control will be returned to the previous application (as defined with the SETUP command). All information about this previous application will be deleted. If no previous application exists, control is returned to the initial application.

If RETURN is issued and no return point is set, the RETURN command will be ignored. Under Natural Security, a LOGOFF command will be executed if RETURN is issued and no return point has been set.

RETURN I

This command causes control to be returned directly to the initial application. This option also causes Natural to delete all definitions of previous applications (except that of the initial application).

RETURN *nn*

This command causes control to be returned to the *nn*th previous application. When this option is used, all information for applications subsequent to the *nn*th application is deleted.

RETURN *

This command will display a list of all return points that are currently set up. On the list you can then select the return point to which you wish to return.

See the SETUP command for further information and examples.

RUN

```
RUN [ REPEAT] [program-name [library-id] ]
```

The RUN command is used to compile and execute a source program. The program may be in the source work area or stored in a Natural library.

The REPEAT parameter indicates that if the program being executed produces multiple screen output, the screens are to be output one after another without intervening prompting messages. When the program terminates, Natural will enter command mode.

program-name

The name of the program to be run. If no program name is specified, Natural will compile and execute the program currently residing in the source work area.

If *program-name* is specified without a library ID, Natural will read the source program into the source work area (overlaying any existing source program), compile, and execute the specified program only if it is stored under the current library ID. If it is not stored under the current library ID, an error message will be issued.

library-id

The library in which the program to be run is contained. If both *program-name* and *library-id* are specified, Natural will retrieve, compile, and execute the specified program only if it is stored under the library ID specified. If it is not stored in the specified library, an error message will be issued.

Under Natural Security, you cannot specify a *library-id*; that is, you can run only programs which are stored in your current library.

SAVE

```
SAVE [object-name [library-id] ]
```

The SAVE command is used to store a source object. The contents of the source work area are not affected.

Note:
The SAVE command will be rejected for an object for which a cataloged object module exists and the parameter RECAT=ON is in effect. In this case, use the STOW command to ensure that the contents of source and object code match.

object-name

The name under which the object is to be saved. If a new object is being saved, an object name must be specified. If an object name is not specified, the object will be saved with the name set by the last command which caused a source object to be read into the source work area (for example, EDIT, READ or RUN).

If an object name is specified without a library ID, Natural will determine if the object is stored under the current library ID. If the object is not stored under the current library ID, the object will be saved using the object name specified. Object names within a library must be unique.

library-id

When you save an object under a different name or save a newly created object, the object will, by default, be stored in the current library. If you wish to store it in another library, you have to specify the desired *library-id* after the *object-name*.

Under Natural Security, you cannot specify a *library-id*; that is, you can store an object only in your current library.

SCAN

```
SCAN
```

Note:
The source work area is used by the SCAN command. Therefore, the object currently in the source work area should be SAVEd or STOWed before SCAN is used.

Any objects except maps, DDMs and data areas can be modified using the SCAN editor. You can also use the full screen editor to modify other lines than scanned lines. When the object is modified, save it and close the editior. You can then continue SCAN processing.

When you issue the SCAN command, a window is displayed. In this window, you can specify the following:

Field	Explanation
Scan Value	The value to be searched for.
Replace Value	The value which is to replace the scan value.
Delete Value	Will cause the scan value to be deleted (YES/NO).
Object Name	The object(s) to be scanned: *blank* or * All objects. *value** All objects whose names begin with *value*.
Object Type	The type of object to be scanned: **P** - Programs **C** - Copycodes **N** - Subprograms **S** - Subroutines **H** - Helproutines **M** - Maps **G** - Global data areas **L** - Local data areas **A** - Parameter data areas **T** - Text **4** - Class ***** - objects of all types
Absolute Scan	Will result in an absolute scan (YES/NO). If Absolute Scan is set to "NO", the scan value must be separated by a blank or specific character.
Library	The ID of the library to be scanned. Default is the current library.
Case Sensitive	**N** - The search will be for the scan value regardless whether it occurs in upper case, lower case, or a mixture of both. **Y** - The search will be for the scan value exactly as you specify it.

Make the desired specifications and press ENTER. A window appears displaying the lines containing the searched for scan value.

```
-----------Search value BONUS in Program ARRAYD--------
  - Commands:
  - 0070    3 BONUS (1:10)
  - 0340  DISPLAY BONUS (1,1)
  - 0350  DISPLAY BONUS (1:5,1)
  - 0360  DISPLAY BONUS (1:5,1:10)
  - 0380  DISPLAY BONUS (#I:#I + 5)
  - 0390  DISPLAY BONUS (#I:#I - 3)
  - 0400  DISPLAY BONUS (#I+2:J-3)
```

SCAN Subcommands

Any desired SCAN subcommand can be entered. Select "Commands:" and press ENTER. Then you can select one of the following commands:

Command	**Function**
Edit	Edit object.
List	List object as it currently appears in the source work area.
Ignore	Ignore the object currently being scanned, do not save any modifications, and continue with next object.
Quit	Terminate SCAN processing.
ESC key	Continue with normal scan processing.

A subcommand can also be invoked by entering its first character.

SCAN Editing Rules

- If the Replace option is used and/or an object is updated in the SCAN editor, the object will always be saved unless an I, Q, or . is specified before the next object is scanned.
- Lines containing PASSWORD=, PASSW=, CIPHER=, or CIPH= will be ignored by the SCAN command.
- The line length of the source object in the SCAN editor is limited to 72.
- If the replace value causes a line to exceed 80 characters, the line will be split automatically.

SCAN under Natural Security

In order to use SCAN in a Natural Security environment, the system commands LIST, EDIT, and READ must be allowed in the current library’s security profile. If the REPLACE option is to be used the system command SAVE must also be allowed.

Under Natural Security, the use of the SCAN command may be disallowed in some libraries.

SCRATCH

```
SCRATCH [object-name ...]
```

The SCRATCH command is used to delete one or more objects - in both source and object form.

As *object-name*, you specify the name(s) of the object(s) to be deleted. You can only delete objects which are stored in your current library.

If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation for the *object-name*.

If you enter the SCRATCH command without an *object-name* or without an *object-name* but with an asterisk, a list of all objects in the current library will be displayed; on the list you may then mark the objects to be deleted.

Notes:
The contents of the source work area is not affected by the SCRATCH command.

If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the SCRATCH command is issued.

SETUP

```
SETUP [application-name] [command-name] [I]
```

The SETUP command is used to define applications to which control is to be returned using the RETURN command. This allows you to easily transfer from one application to another during a Natural session.

If SETUP is issued without any parameters, a menu will be displayed for the purpose of entering the command information.

application-name

The name of the application to which control is to be returned. A maximum of 8 characters may be used (A8).

If *application-name* is blank, a LOGON command will not be issued. This permits multiple return points within the same application.

If *application-name* is "*", the current value of the Natural system variable *LIBRARY-ID (that is, at the time SETUP is issued) is used to create the LOGON command when RETURN is issued.

command-name

The name of the command which is to be executed when control is returned to the application. A maximum of 60 characters may be used (A60).

If *command-name* is blank, no command will be issued after the LOGON.

If *command-name* is "*", the current value of the Natural system variable *STARTUP (that is, at the time SETUP is issued) is used as the startup command when RETURN is issued.

I Option

If the "I" option is specified, all return points defined with previous SETUP commands will be deleted and the application specified with "SETUP I" will be defined as the new initial application.

SETUP/RETURN Example:

1. User starts Natural session (default application = "APPL1")
   Return point APPL1 is defined on level 1.
2. User issues command "LOGON APPL2"
3. User executes a program which stacks two commands:
   SETUP *,MENU (establish return point)
   LOGON APPL3 (go to another application)
   Return point APPL2, STARTUP MENU is defined on level 2.
4. User issues command "LOGON APPL4" (user selects other application)
5. User presses an F key which has the value "RETURN"
   Natural will issue for the user:
   LOGON APPL2
   MENU
   Return to APPL2, delete level 2.
6. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL5

Return point APPL2, STARTUP MENU is defined on level 2.

7. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL6
   Return point APPL5, STARTUP MENU is defined on level 3.
8. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL7
   Return point APPL6, STARTUP MENU is defined on level 4.
9. User executes a program which stacks:
   SETUP *,MENU
   LOGON APPL8
   Return point APPL7, STARTUP MENU is defined on level 5.
10. User executes a program which stacks:
    SETUP *,MENU
    LOGON APPL9
    Return point APPL8, STARTUP MENU is defined on level 6.
11. User issues command "RETURN 2" (return 2 levels back)
    Natural will return user to APPL7, since that was the 2nd previous session (all information for APPL8 is now lost).
    Level 6 (APPL8) is deleted, level 5 (APPL7) is activated and level deleted.
12. User issues command "RETURN"
    Level 4 (APPL6) is activated, level deleted.
    Natural will return user to APPL6, since that was the session previous to APPL7.
13. User issues command "RETURN"
    Level 3 (APPL5) is activated, level deleted.
    Natural will return user to APPL5, since that was the session previous to APPL6.
14. User issues command "RETURN I"
    Level 2 (APPL2) is deleted, level 1 (APPL1) is activated.

STOW

```
STOW [object-name [library-id] ]
```

The STOW command is used to store an object - in both source and object form.

object-name

The name of the object to be stored.

An object name need not be specified if an object is being stored that was read from a library and is being stored under the same name.

If the object is being stored under a different name, the new object name must be specified.

If a new object is being stored, an object name must be specified. The new object name must not already exist in either source or object form.

library-id

When you store an object under a different name or store a newly created object, the object will, by default, be stored in the current library. If you wish to store it in another library, you have to specify the desired *library-id* after the *object-name*.

Under Natural Security, you cannot specify a *library-id*; that is, you can store an object only in your current library.

Note:
If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the STOW command is issued.

STRUCT

```
STRUCT [(n)]
```

The STRUCT command performs structural indentation of a source program.

The following types of statements are affected by the STRUCT command:

- processing loops (READ, FIND, FOR, etc.),
- conditional statement blocks (AT BREAK, IF, DECIDE FOR, etc.),
- DO/DOEND statement blocks,
- DEFINE DATA blocks,
- inline subroutines.

With this function, you can have a source program indented so that the indentation of source-code lines reflects the structure of the program.

Note:
Indentation is performed differently for a reporting-mode program than for a structured-mode program.

Partial Indentation

You can exclude sections of your program source from structural indentation by using the special statements "/*STRUCT OFF" and "/*STRUCT ON". These must be entered at the beginning of a source-code line. The source-code lines between these two statements will remain as they are when you execute the Generate function.

Example of Structural Indentation:

Program before being structurally indented:

```
0010 DEFINE DATA LOCAL
0020 1 EMPL VIEW OF EMPLOYEES
0030 2 PERSONNEL-ID
0040 2 FULL-NAME
0050 3 FIRST-NAME
0060 3 NAME
0070 1 VEHI VIEW OF VEHICLES
0080 2 PERSONNEL-ID
0090 2 MAKE
0100 END-DEFINE
0110 FIND EMPL WITH NAME = ’ADKINSON’
0120 IF NO RECORDS FOUND
0130 WRITE ’NO RECORD FOUND’
0140 END-NOREC
0150 FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
0160 DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
0170 END-FIND
0180 END-FIND
0190 END
```

The same program after the function Generate Structured Source has been applied to it:

```
0010 DEFINE DATA LOCAL
0020 1 EMPL VIEW OF EMPLOYEES
0030   2 PERSONNEL-ID
0040   2 FULL-NAME
0050     3 FIRST-NAME
0060     3 NAME
0070 1 VEHI VIEW OF VEHICLES
0080   2 PERSONNEL-ID
0090   2 MAKE
0100 END-DEFINE
0110 FIND EMPL WITH NAME = ’ADKINSON’
0120   IF NO RECORDS FOUND
0130     WRITE ’NO RECORD FOUND’
0140   END-NOREC
0150   FIND (1) VEHI WITH PERSONNEL-ID = EMPL.PERSONNEL-ID
0160     DISPLAY EMPL.PERSONNEL-ID FULL-NAME MAKE
0170   END-FIND
0180 END-FIND
0190 END
```

The parameter (n) may be supplied to specify the number of spaces used for identation. If (n) is unspecified, identation is set to 2.

Example:

```
STRUCT (5)
```

SYSDDM

```
SYSDDM
```

This command is used to invoke the SYSDDM utility, see also section DDM Services.

SYSERR

```
SYSERR
```

This command is used to invoke the SYSERR utility. This utility is described in the Natural SYSERR Utility documentation.

SYSEXT

```
SYSEXT
```

With the SYSEXT command, you display a list of all available Natural user exits.
For each listed user exit, you can perform the following functions:

- Edit example
- List example
- Run example
- Execute example
- List documentation
- List keywords

SYSFILE

With SYSFILE you get information about:

- Reports
- Logical Devices
- Defined Physical Devices
- Defined Printer Profiles
- Defined Workfiles

SYSMAIN

```
SYSMAIN
```

This command is used to invoke the SYSMAIN utility. This utility is described in the section Library Maintenance.

SYSNCP

```
SYSNCP
```

This command is used to invoke the SYSNCP utility. This utility is described in the Natural SYSNCP Utility documentation.

SYSOBJH

This command is used to invoke the SYSOBJH utility.
Plese read the section the SYSOBJH Utility for further information.

SYSPROD

```
SYSPROD
```

With the SYSPROD command, you can find out which products are installed at your Natural site: that is, Natural itself, Natural Selectable Units, and products running under Natural.

When you enter the command, a window will be displayed, listing the following information for each product installed:

- the product name,
- the product version,
- the system maintenance (SM) level,
- the installation date.

For some of the products listed, you can obtain additional information by marking them with a command. For a list of available commands, enter a "?" in the "Cmd" column of the SYSPROD window.

SYSPROF

```
SYSPROF
```

With the SYSPROF command, you can display the current definitions of the Natural system files. For each file, you can display:

- the file name,
- the database ID,
- the file number,
- the database type.

SYSTRANS

```
SYSTRANS
```

This command is used to invoke the SYSTRANS utility.
Read the Natural SYSTRANS Utility documentation for further information.

SYSUNLD

```
SYSUNLD
```

This command is used to invoke the library SYSUNLD. SYSUNLD contains the utilities NATUNLD and NATLOAD.
These utilities are described in the Natural NATUNLD/NATLOAD Utilities documentation.

TECH

```
TECH
```

With the TECH command, you can display technical information about your Natural session.

The following information is displayed:

- user ID
- library ID
- Natural version and SM level
- startup transaction
- Natural Security indicator
- operating system name and version
- machine class
- hardware
- terminal type
- last command issued
- information on the last error that occurred:
    - error number
    - number of the line in which the error occurred
    - name, type and level of the object that caused the error
    - name, database ID and file number of the library containing the object
    - error class
      (system = error issued by Natural; user = error issued by user application)
    - error type (runtime, syntax, command execution, session termination, program termination, remote procedure call)
    - date and time at which the error occurred
    - name of the error transaction
- names, database IDs and file numbers of all currently active steplibs
- names, types and levels of the currently active programming object and all objects on higher levels, as well as the line numbers of the statements invoking the subordinate programming objects.

Note:
To display this information from any point in an application, you can use the terminal command %<TECH.

UNCATALOG

```
UNCATALOG [object-name]...
```

The UNCATALOG command is used to delete one or more object modules.

As *object-name*, you specify the name of the object to be deleted. You can only delete objects which are stored in the library to which you are currently logged on.

If more than one object is to be deleted, the *object-names* must be separated by one or more blanks (or the currently defined delimiter character).

If you wish to delete all objects whose names begin with a specific string of characters, use asterisk notation for the *object-name*.

If you enter the UNCATALOG command without an *object-name* or without an *object-name* but with an asterisk, a list of all cataloged objects in the current library will be displayed; on the list, you can then mark the object(s) to be deleted.

The contents of the source work area is not affected by the UNCATALOG command.

Note:
If an FDIC system file is specified in the parameter file which is not valid, Natural will display an appropriate error message when the UNCATALOG command is issued.

UNREGISTER

This command is used in conjunction with NaturalX. It is described in the NaturalX Documentation.

UPDATE

$$\text{UPDATE} \left\{ \begin{matrix} \text{ON} \\ \text{OFF} \end{matrix} \right\}$$

The UPDATE command is used to prevent (or allow) database updating.

The UPDATE ON command allows updating.

The UPDATE OFF command prevents updating which would normally be performed as a result of an UPDATE, STORE, or DELETE statement. Programs containing these statements will execute normally but no modification of the database will occur. When an update operation is encountered, a message will be displayed instead of a database update being performed.

When the system command CHECK is used with UPDATE OFF, an error message is displayed.

The UPDATE command has no effect on other Natural system commands.

XREF

```
       [ ON    ]
       [ OFF   ]
  XREF [ FORCE ]
       [ DOC   ]
       [ ?     ]
```

This command is only available if Predict has been installed.

This command controls the usage of the Predict function "active cross-references".

The active cross-reference facility automatically creates documentation in the Data Dictionary about Natural objects. These objects include programs, subprograms, subroutines, helproutines, maps, data areas, database views, database fields, user-defined variables, processing rules, error numbers, work files, printers, classes, events, copycodes, and retained ISN sets.

The active cross-reference is created when a Natural object is cataloged.

To look at cross-reference data, you use the XREF option of the system command LIST.

For further information on active cross-references, see the Predict documentation.

The following command options are available:

XREF

If you enter the XREF command without parameters, a menu is displayed where you specify the desired option.

XREF ON

This command enables the generation of active cross-reference data. Cross-reference data are stored in the respective Predict entries each time a Natural object is cataloged.

XREF OFF

This command disables the generation of active cross-reference data. Existing cross-reference data for the object being cataloged are deleted.

XREF FORCE

The object can only be cataloged if a Predict entry exists for it. When the object is cataloged, its cross-reference data will be stored in Predict.

If no Predict entry exists, the object cannot be cataloged.

XREF DOC

The object can only be cataloged if a Predict entry exists for it. However, when the object is cataloged, no cross-reference data will be generated and existing cross-reference data for the object will be deleted.

If no Predict entry exists, the object cannot be cataloged.

XREF ?

Help function for the XREF command.

Natural Security, Considerations

If Natural Security, is installed, the setting for XREF may be set for each library in the library security profile. Depending on the security profile, some options of the XREF command may not be available to you.